#3

PCT/DE 00/00623

BUNDESREPUBLIK DEUTSCHLAND

EU 09/936-385

DE00/00623

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

Bescheinigung

Die Siemens Aktiengesellschaft in München/Deutschland hat eine Patentanmeldung unter der Bezeichnung

"Verfahren zur Nachrichtenübertragung zwischen einer einem ersten Prozeß zugewiesenen Clientinstanz und wenigstens einer mindestens einem weiteren Prozeß zugewiesenen Serverinstanz innerhalb eines verteilten Systems"

am 09. März 1999 beim Deutschen Patent- und Markenamt eingereicht.

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

Die Anmeldung hat im Deutschen Patent- und Markenamt vorläufig das Symbol G 06 F 15/163 der Internationalen Patentklassifikation erhalten.

München, den 17. April 2000

Deutsches Patent- und Markenamt

Der Präsident

Im Auftrag

Brand

Aktenzeichen: 199 10 345.3

A 9161
06.90
11/98

Beschreibung

Verfahren zur Nachrichtenübertragung zwischen einer einem ersten Prozeß zugewiesenen Clientinstanz und wenigstens einer mindestens einem weiteren Prozeß zugewiesenen Serverinstanz innerhalb eines verteilten Systems

Die Erfindung betrifft ein Verfahren zur Nachrichtenübertragung zwischen einer einem ersten Prozeß zugewiesenen Clientinstanz und wenigstens einer mindestens einem weiteren Prozeß zugewiesenen Serverinstanz innerhalb eines verteilten Systems.

Verteilte Systeme spielen vorzugsweise in heutigen Telekommunikationssystemen, die in der Regel Multiprozessorsysteme sind, eine besondere Rolle. Ein verteiltes System ist insbesondere dadurch charakterisiert, daß Prozesse jeweils unterschiedlichen Prozessoren zugeteilt werden können, wobei sich die Prozessoren gegebenfalls auf örtlich getrennten Plattformen im verteilten System befinden können. Einer der wichtigsten Aspekte bei der Kommunikation zwischen verschiedenen Prozessen eines verteilten Systems ist dabei die Plattform-Transparenz. Damit ist gemeint, daß ein Prozeß, der eine Nachricht an einen anderen Prozeß senden will, die Plattform, auf der der andere Prozeß gerade abläuft, nicht kennen muß. Solch ein komplexes verteiltes System muß heutzutage noch vielen weiteren Anforderungen genügen. Es muß sich unter anderem als äußerst zuverlässig, möglichst flexibel sowie offen für Anpassungen und Erweiterungen erweisen. Die Software eines derartigen komplexen verteilten Systems soll daher hochgradig modular mit fest definierten offenen Schnittsstellen nach außen gestaltet sein, damit die einzelnen Module der Software leicht anpassungsfähig und vor allem wiederverwendbar sind.

Um den genannten Anforderungen insbesondere der Wiederverwendbarkeit von Software einigermaßen gerecht zu werden, wird

die Software zu einem solchen verteilten System mit Hilfe objektorientierter Entwurfmethoden und objektorientierter Programmierung erstellt. Jedoch ist die in verteilten Systemen notwendige Zuordnung von Objekten untereinander, die meist unterschiedlichen und gegebenfalls nebenläufigen Prozessen zugewiesen werden, nicht zufriedenstellend gelöst. Teilweise muß sogar ein rein objektorientierter Systementwurf in herkömmliche prozedurale Programmiertechniken aufgebrochen werden, wodurch der mit der Objektorientierung erreichte Effekt der Wiederverwendung von Programmteilen mehr oder weniger verloren geht.

Derzeit werden bei der Einführung von Nebenläufigkeit und Parallelverarbeitung in die Welt der Objekte folgende bekannte Ansätze diskutiert:

- Implizite Nebenläufigkeit: Bei der Implementierung von implizierter Nebenläufigkeit gibt es zwei Möglichkeiten:

  - Passive Objekte: Ein asynchroner Nachrichtenaustausch wird in einen sequenziellen synchronen Methoden- bzw. Prozeduraufruf umgewandelt. Hierbei wird die Parallelverarbeitung der miteinander kommunizierenden Objekte sehr eingeschränkt.
  - Aktive Objekte: Für jedes Objekt wird ein Prozeß gestartet. Dieses Vorgehen führt zu einem hohen Ressourcenverbrauch und ist deshalb nur mit einer begrenzten Anzahl von Objekten realisierbar.

- Explizite Nebenläufigkeit: Hierbei wird entweder eine Gruppe von Objekten (objektbezogen), wie in einem Artikel von A. Coutts, J. M. Edwards, Model-Driven Distributed Systems, IEEE Concurrency, Juli 1997, S. 55-63 beschrieben, oder mehrere Ereignisse in einer Ablaufsequenz (aufgabenbezogen), wie in einem Artikel von M. Awad, J. Ziegler, A Practical Approach to the Design of Concurrency in Object-Oriented Systems, Software - Practice and Experience, Sep-

tember 1997, Vol. 27(9), S. 1013-1034 erläutert, einem Prozeß zugewiesen. Bei Betrachtung der rechten Hälfte der Figur 3 im genannten Artikel von Awad/Ziegler und der Figur 5 im genannten Artikel von Coutts/Edwards ist an den Schnittstellen zwischen den Objekten, die teilweise gleichzeitig Schnittstellen zwischen den Prozessen darstellen, zu erkennen, daß die Kommunkation zwischen den Objekten sowohl durch synchrone Methodenaufrufe als auch durch Interprozeßkommunikation in Form einer asynchronen Nachrichtenweitergabe erfolgt. Eine derartige Festlegung der Kommunikationsart an den Schnittstellen von Objekten hat den Nachteil, daß die Wiederverwendbarkeit und die Wartbarkeit der Objekte erheblich erschwert wird.

Insbesondere im Zusammenhang mit der Kommunikation zwischen verschiedenen Objekten eines verteilten Systems, auch Instanzen genannt, die untereinander in der Regel in einem sogenannten Client/Server-Verhältnis stehen und die verschiedenen Prozessen zugewiesen sind, ist die vorstehend erläuterte Vorgehensweise hinsichtlich der in einem solchen komplexen System erwünschten Wiederverwendbarkeit und Wartbarkeit eine sehr ungünstige Lösung.

Die Aufgabe der Erfindung besteht daher darin, ein Verfahren zur Nachrichtenübertragung zwischen sogenannten jeweils unterschiedlichen Prozessen zugewiesenen Client- und Serverinstanzen eines verteilten Systems dahingehend auszugestalten, daß bezüglich der Implementierung des Verfahrens eine möglichst hohe Wiederverwendbarkeit gegeben ist und zugleich die Wartbarkeit möglichst erleichtert wird.

Diese Aufgabe wird durch die im Anspruch 1 angegebenen Merkmale gelöst. Weitere Ausgestaltungen der Erfindung sind in Unteransprüchen gekennzeichnet.

Erfindungsgemäß wird dies dadurch erreicht, daß zur Nachrichtenübertragung zwischen einer einem ersten Prozeß zugewiese-

nen Clientinstanz und wenigstens einer mindestens einem weiteren Prozeß zugewiesenen Serverinstanz innerhalb eines verteilten Systems zusätzlich als gegenseitige Kommunikationspartner vorgesehene Partnerinstanzen eingesetzt werden. Eine den ersten Prozeß enthaltende erste Instanz der Partnerinstanzen wählt nach Empfang einer von der Clientinstanz an wenigstens eine Serverinstanz gerichtete Nachricht mindestens eine geeignete den mindestens einen weiteren Prozeß enthaltende weitere Instanz der Partnerinstanzen zur Nachrichtenannahme und -weitergabe aus. Die mindestens einen weiteren Prozeß enthaltende weitere Instanz leitet diese Nachricht zu wenigstens einer von ihr adressierten Serverinstanz weiter und erhält gegebenfalls von der wenigstens einen Serverinstanz eine Nachricht zur Weiterleitung über die den ersten Prozeß enthaltende erste Instanz an die Clientinstanz.

Auf diese Weise wird die Festlegung der Kommunikationsart zwischen der Clientinstanz und der mindestens einen Serverinstanz in die einen Prozeß enthaltenden, als gegenseitige Kommunikationspartner vorgesehenen Partnerinstanzen verlagert. So werden die Nachrichten zwischen der Clientinstanz und der den ersten Prozeß enthaltenden ersten Instanz sowie zwischen der mindestens einen Serverinstanz und der mindestens einen weiteren Prozeß enthaltenden weiteren Instanz synchron z.B. durch Prozedur- bzw. Methodenaufruf übertragen. Die Nachrichtenübertragung zwischen einer den ersten Prozeß enthaltenden ersten Instanz und einer mindestens einen weiteren Prozeß enthaltenden weiteren Instanz kann dann entkoppelt von den Kommunikationsschnittstellen der Clientinstanz und mindestens einen Serverinstanz asynchron oder synchron erfolgen. Dadurch wird eine maximale Wiederverwendbarkeit vorwiegend bezüglich der Implementierung der Client- und der mindestens einen Serverinstanz erreicht. Die Wartbarkeit wird ebenfalls erheblich verbessert dadurch, daß allenfalls die Kommunikationsschnittstellen zwischen der den ersten Prozeß enthaltenden ersten Instanz und der mindestens einen weiteren Prozeß enthaltenden weiteren Instanz angepaßt werden müssen, jedoch die Kommuni-

kationsschnittstellen der Client- und der mindestens einen Severinstanz unberührt bleiben.

Eine weitere vorteilhafte Ausgestaltung der Erfindung sieht vor, daß die den ersten Prozeß enthaltenden erste Instanz die Auswahl der mindestens einen weiteren Prozeß enthaltenden weiteren Instanz anhand einer Zuordnungstabelle trifft. In dieser Zuordnungstabelle ist die Art der von der Clientinstanz aussendbaren Nachrichten und die Adresse der mindestens einen weiteren Prozeß enthaltenden weiteren Instanz eingetragen. Eine Zuordnungstabelle hat den Vorteil, daß ihr Inhalt jederzeit änderbar ist, und der den ersten Prozeß enthaltenden ersten Instanz eine schnelle Auswahl ermöglicht.

Gemäß einer zweckmäßigen Weiterbildung der Erfindung ist die durch die den ersten Prozeß enthaltende ersten Instanz getroffene Auswahl dynamisch in Abhängigkeit von der Systemauslastung änderbar. Dadurch können Systemabstürze sowie Verklemmungen bei der Zuteilung der Prozesse auf die Prozessoren vermieden werden.

Eine weitere Ausgestaltung der Erfindung betrifft den Spezialfall, daß der erste Prozeß und der mindestens eine weitere Prozeß zusammenfallen. In diesem Fall sind die den ersten Prozeß enthaltende erste Instanz und die den mindestens einen weiteren Prozeß enthaltende weitere Instanz in einer Instanz vereinigt. Dadurch kann das erfindungsgemäße Verfahren ohne Anpassungen auf diesen Spezialfall angewendet werden.

Eine weitere nützliche Ausgestaltung der Erfindung ist in der Art der Implementierung zu sehen. So können sämtliche Instanzen (Client-, Server- , die den ersten Prozeß enthaltende Instanz und Partnerinstanz) in Form von Objekten implementiert werden, deren Struktur durch Objektklassen festgelegt wird. So weisen die den ersten Prozeß enthaltende erste Instanz und die mindestens einen weiteren Prozeß enthaltende weitere Instanz vorzugsweise jeweils die Struktur einer gemeinsamen Ob-

jektklasse auf. Auf diese Weise werden die Grundsätze der rein objektorientierten Programmierung ausgenutzt, wodurch ein hoher Grad an Modularität, eine hohe Wiederverwendbarkeit und Wartbarkeit erreicht wird.

Eine weitere Ausgestaltung der Erfindung ist in einer sehr zweckmäßigen Verwendung des erfindungsgemäßen Verfahrens auf ein Fernsprechvermittlungssytem zu sehen. Demnach kommen alle vorstehend erwähnten Vorteile auch im Zusammenhang mit einem Fernsprechvermittlungssystem zum Tragen.

Nachstehend wird ein Ausführungsbeispiel der Erfindung unter Bezugnahme auf eine Zeichnung näher beschrieben.

In der Zeichnung zeigen:

Figur 1 ein beispielhaftes Ablaufdiagramm des erfindungsgemäßen Verfahrens,

Figur 2 ein Anwendungsbeispiel im Bereich einer System-Alarmierung in einem Telekommunikationssystem wie z.B. einem Fernsprechvermittlungssystem

Eine Legende zu den Figuren ist im Anhang am Ende der Beschreibung zu finden.

Figur 1 beschreibt in einem Ablaufdiagramm die Nachrichtenübertragung zwischen einer einem ersten Prozeß zugewiesenen Clientinstanz und einer einem weiteren Prozeß zugewiesenen Serverinstanz. Die Instanzen Client, Server , die den ersten Prozeß enthaltende erste Instanz und die den mindestens einen weiteren Prozeß enthaltende weitere Instanz sowie die Aktion, die von der Serverinstanz ausgeführt wird, werden in Form von Objekten mit Kästchen dargestellt. So entpricht das Objekt Client einer Clientinstanz, das Objekt Server einer Serverinstanz, das Objekt ObjectHandler1 einer den ersten Prozeß enthaltenden ersten aktiven Instanz der als gegenseitige Kommu-

nikationspartner vorgesehenen Partnerinstanzen, das Objekt ObjectHandler2 einer den weiteren Prozeß enthaltenden weiteren aktiven Instanz der Partnerinstanzen, das Objekt Action einer Aktion und das Objekt Confirm Action einer Rückmeldungsaktion auf eine angeforderte Aktion. Die aktiven Instanzen, die die jeweiligen Prozesse enthalten, sind hierbei durch Kästchen mit fett gezeichneten Linien gekennzeichnet. Die Art der Aktion wird erst beim Aufruf des speziellen Objekts Action bestimmt.

Im Falle einer vom Client angeforderten vom Server auszuführenden Aktion mit Rückmeldung läuft das Verfahren beispielsweise wie folgt ab:

Ein Client fordert vom Server eine Aktion an, auf die eine Rückmeldung erfolgen soll. Der Client ruft die Aktion auf und muß nicht wissen, welcher Prozeß bzw. auf welcher Prozessor-Plattform die Aktion ausgeführt werden soll. Der Objecthandler1 stellt den Client dafür die Aufrufprozedur invoke_action bereit. Nach dem Aufruf der Prozedur invoke_action, in der objektorientierten Programmierung auch Methode genannt, wird dem ObjectHandler1 eine eindeutige Nummer (get handle number) zugeordnet und es wird ein Zeitnehmer gestartet (start timer), der bei nicht rechtzeitigem Eintreffen der Rückmeldung eine Fehlerbehandlung auslöst. Danach sucht der ObjectHandler1 nach einer als Kommunikationspartner vorgesehenen Partnerinstanz z.B. ObjectHandler2 (find target ObjectHandler), der der Aktion abhängig von der Art der Aktion zugeordnet ist, und übermittelt die Nachricht der Aktionsanforderung action_request an den ObjectHandler2. Der ObjectHandler2 nimmt die Nachricht entgegen, speichert die Adresse seines Kommunikationspartners Objecthandler1 (store communication partner) zusammen mit der dem ObjectHandler1 eindeutig zugeordneten Nummer und führt die Prozedur des Objekts Action aus (execute). Das Objekt Action veranlaßt daraufhin den vom Client adressierten Server zur Ausführung der Aktion durch den Prozeduraufruf action. Nach der Ausführung der Aktion sendet der

Server in analoger Weise indirekt eine Rückmeldung zum Client zurück. Demnach laufen folgende Prozeduraufrufe, Nachrichtenübertragungen und Aktionen vom Server in Richtung zum Client ab. Prozeduraufruf Invoke_action, lösche Adresse des Kommunikationspartners sowie Übertragung der Aktionsanforderungsnachricht für die Rückmeldung action_request vom ObjectHandler2 zum ObjectHandler1, der dem ObjectHandler2 aufgrund der zugeordneten Nummer bekannt ist, ObjectHandler1 löscht die zugeordnete Nummer (release handle number) und stoppt den Zeitnehmer (stop timer), zur Übermittlung der Rückmeldung ruft Objecthandler1 die Prozedur execute des Objekts Confirm Action auf und zuletzt führt Objekt Conform Action die Prozedur confirm_action des Client aus.

Im Falle einer vom Client angeforderten Aktion des Servers ohne Rückmeldung läuft das erfindungsgemäße Verfahren der Nachrichtenübertragung vom Client zum Server in ähnlicher Weise wie vorstehend beschrieben ab. Es entfallen die Ablaufschritte get handle number, start timer, store communication partner und die Schritte bezüglich der Rückmeldung vom Server in Richtung zum Client.

Im Falle eines sogenannten Broadcasts, d.h. ein Client fordert von mehreren Servern eine Aktion an, gibt es verschiedene Möglichkeiten:

- Wenn die vom Client adressierten Server einem gemeinsamen Prozeß zugewiesen sind, wird der ObjectHandler1 die action_request-Nachricht entweder an einen ObjectHandler2 weitergeben und der ObjectHandler2 sorgt dafür, daß die Aktion von mehreren Servern ausgeführt wird, oder der ObjectHandler1 sendet mehrere action_request-Nachrichten an mehrere den Server-Prozeß enthaltende ObjectHandler2, die jeweils die Server zur Ausführung der Aktion veranlassen. Auch ist eine Kombination aus beiden genannten Varianten möglich.
- Wenn die vom Client adressierten Server unterschiedlichen Prozessen zugewiesen sind, wird der ObjectHandler1 jeweils

eine action_request-Nachricht an die die unterschiedlichen Prozessen enthaltenden ObjectHandler2 senden und die ObjectHandler2 veranlassen jeweils die Server zur Ausführung der Aktion.

Auch hier sind sämtliche Kombinationen der erwähnten Möglichkeiten denkbar.

Üblicherweise sind in einem verteilten System mehrere Aktionen auszuführen, so daß selbstverständlich jeder Server auch als Client und jeder Client auch als Server agieren kann sowie in einem Objekt Client- und Serverfunktion vereint sein können.

Die vorteilhafte Entkopplung der Prozeßschnittstellen von den Objektschnittstellen des Client und des Servers ist daran zu erkennen, daß die Kommunikation zwischen den Client und dem Server synchron durch Prozedur- bzw. Methodenaufrufe realisiert wird und nur die Nachrichtenübergabe zwischen den ObjectHandler1 und ObjectHandler2 gegebenfalls asynchron über die Prozeßgrenzen hinweg durchgeführt wird.

In dem Spezialfall, daß der Client und der Server, die sich beispielsweise auf einer gemeinsamen Plattform befinden, demselben Prozeß zugewiesen werden können, sind die Objekte ObjectHandler1 und ObjectHandler2 zu einem einzigen Objekt vereinigt. Gemäß der Figur 1 sendet der ObjectHandler1 die action_request-Nachricht in diesem Fall an sich selbst.

Figur 2 zeigt ein Anwendungsbeispiel im Bereich einer System-Alarmierung in einem Telekommunikationssystem z.B. einem Fernsprechvermittlungssystem.

Bei einer System-Alarmierung gibt es beispielsweise folgende Objekte, die zugleich als Client und Server agieren und untereinander unterschiedliche Aktionen anfordern können. Au-

ßerdem können sich die Objekte auf verschiedenen Plattformen befinden.

Ein Objekt-Alarmbilanz-Monitor (ABM) hat die Aufgabe, eine Alarmbilanz über alle Alarme der von ihm überwachten alarmierbaren Instanzen (AMOI) zu ziehen. Um die Alarmbilanz ziehen zu können, benötigt der Alarmbilanz-Monitor mindestens ein sogenanntes SIBS-Objekt, das sich auf einer Prozessorplattform befindet und ihm eine gesammelte Information bezüglich der überwachten alarmierbaren Instanzen liefert.

Die Kästchen stellen die Objekte Caller, AMOI (AlarmManagedObjectInstance), SIBS (SiteBalanceSupply) und ABM (AlarmBalanceMonitor) dar. Durch die Pfeile, deren Art in der Legende im Anhang nicht aufgeführt ist, wird die Nachrichtenübertragung gegebenfalls über Prozeßgrenzen hinweg zwischen den Objekten angedeutet. Die Nachrichtenübertragung entspricht dabei der in der Figur 1 beschriebenen Nachrichtenübertragung zwischen Client und Server. So kann beispielsweise das Caller-Objekt als Client und das AMOI-Objekt als Server agieren. Entsprechendes gilt auch für die übrigen Objekte AMOI und SIBS sowie SIBS und ABM.

Nach einem System-Alarm-Aufruf set_alarm wird beispielsweise folgender Ablauf von Aktionen ausgelöst:

- Set_alarm: Eine überwachte alarmierbare Instanz AMOI erhält von einem Aufrufer Caller einen neuen Alarm, prüft die den Alarm bestimmenden Parameter (check_params) und kreiert eine neue Alarminstanz (create contained alarm).
- Confirm: Eine Rückmeldung von der Instanz AMOI an die Instanz Caller nach dem System-Alarm-Aufruf set_alarm.
- Balance SIBS: Mindestens ein Serverobjekt SIBS wird aufgefordert, die erhaltenen für die Alarmbilanz notwendigen Informationen zu sammeln (accumulate alarm status of all associated AMOI).
- Balance ABM: Danach wird das Serverobjekt ABM aufgefordert, die von den mindestens einen SIBS-Objekt erhaltenen

Informationen für die Alarmbilanz zu sammeln (accumulate alarm status of all associated SIBS) .

Da die Aktionen über Prozeßgrenzen hinweg angefordert werden, werden die Nachrichten von einem Objekt zu einem weiteren Objekt über eine aktive erste Instanz und über eine aktive weitere Partnerinstanz übertragen wie z.B. über den ObjectHandler1 und über den ObjectHandler2 aus Figur 1, die beide in der Figur 2 nicht dargestellt sind.

Die durch den ObjectHandler1 getroffende Auswahl des Objekt-Handlers 2 kann anhand einer Zuordnungstabelle vorgenommen werden. Die Zuordnungstabelle sieht beispielsweise wie folgt aus:

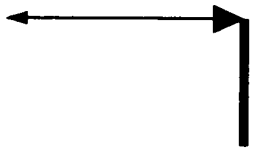
Aktion	ObjectHandler	Rückmeldung
Set_alarm AMOI	Auf AMOI-Plattform	ja
Balance SIBS	Auf SIBS-Plattform	nein
Balance ABM	Auf Hauptplattform	nein

Sofern eine bestimmte Aktion von unterschiedlichen Serverobjekten ausgeführt werden kann, kann die Zuordnung des ObjectHandler2 abhängig von der Systemauslastung geändert werden.

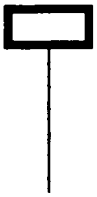
ANHANG

Legende zu den Figuren:

= Instanz



= Prozeduraufruf mit Ausführung (synchron)



= aktive Instanz, die einen Prozess enthält

= Senden einer Nachricht (asynchron)I

= Instanz wartet auf Rückmeldung

Patentansprüche

1. Verfahren zur Nachrichtenübertragung zwischen einer einem ersten Prozeß zugewiesenen Clientinstanz (Client) und wenigstens einer mindestens einem weiteren Prozeß zugewiesenen Serverinstanz (Server) innerhalb eines verteilten Systems, wobei eine den ersten Prozeß enthaltende erste Instanz (ObjectHandler1) von als gegenseitige Kommunikationspartner vorgesehenen Partnerinstanzen nach Empfang einer von der Clientinstanz an wenigstens eine Serverinstanz gerichteten Nachricht mindestens eine geeignete den mindestens einen weiteren Prozeß enthaltende weitere Instanz (ObjectHandler2) der Partnerinstanzen zur Nachrichtenannahme und -weitergabe auswählt, die die Nachricht zu wenigstens einer von ihr adressierten Serverinstanz weiterleitet und gegebenfalls von der wenigstens einen Serverinstanz eine Nachricht zur Weiterleitung über die den ersten Prozeß enthaltende erste Instanz an die Clientinstanz erhält.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß die den ersten Prozeß enthaltende erste Instanz die Auswahl der mindestens einen weiteren Prozeß enthaltenden weiteren Instanz anhand einer Zuordnungstabelle zwischen der Art der von der Clientinstanz aussendbaren Nachrichten und der Adresse der mindestens einen weiteren Prozeß enthaltenden weiteren Instanz trifft.

3. Verfahren nach Anspruch 2, dadurch gekennzeichnet, daß die durch die den ersten Prozeß enthaltende erste Instanz getroffene Auswahl dynamisch in Abhängigkeit von der Systemauslastung änderbar ist.

4. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß dann, wenn der erste Prozeß und der mindestens eine weitere Prozeß zusammenfallen, die den ersten Prozeß enthaltende erste Instanz und

die den mindestens einen weiteren Prozeß enthaltende weitere Instanz in einer Instanz vereinigt sind.

5. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß sämtliche Instanzen in Form von Objekten implementiert sind, deren Struktur durch Objektklassen festgelegt wird.

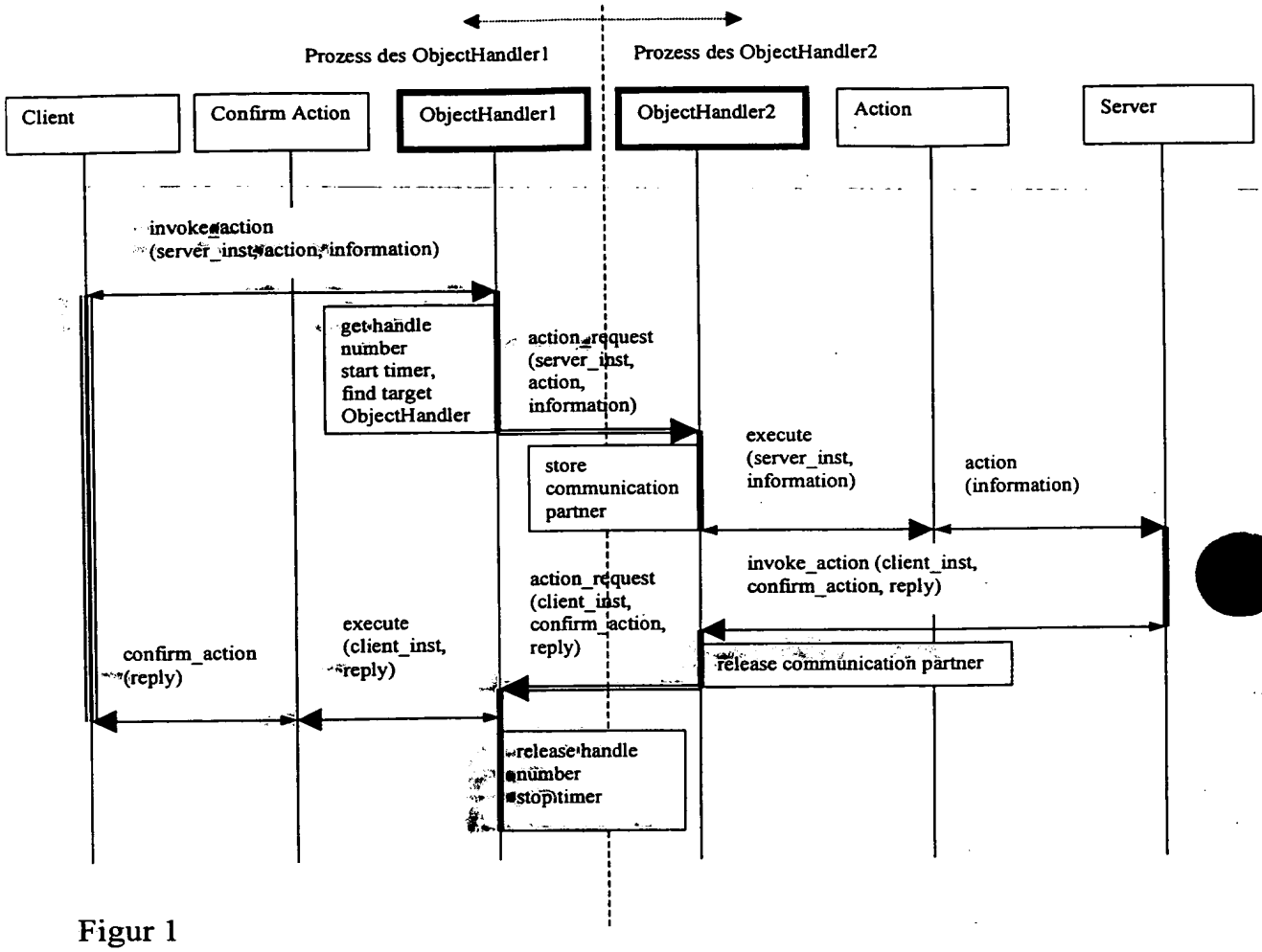
6. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß es auf ein Fernsprechvermittlungssystem angewendet wird.

Zusammenfassung

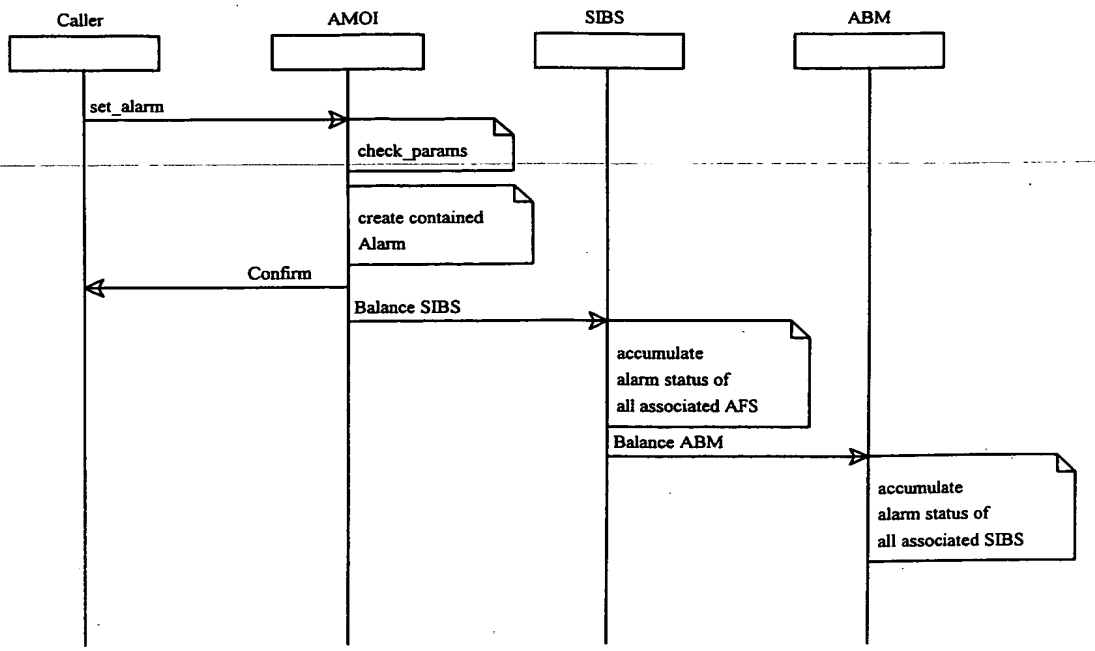
Verfahren zur Nachrichtenübertragung zwischen einer einem ersten Prozeß zugewiesenen Clientinstanz und wenigstens einer mindestens einem weiteren Prozeß zugewiesenen Serverinstanz innerhalb eines verteilten Systems

Eine einen ersten Prozeß enthaltende erste Instanz (ObjectHandler1) von als gegenseitige Kommunikationspartner vorgesehenen Partnerinstanzen wählt nach Empfang einer von der Clientinstanz (Client) an wenigstens eine Serverinstanz (Server) gerichtete Nachricht mindestens eine geeignete den mindestens einen weiteren Prozeß enthaltenden weitere Instanz (ObjectHandler2) der Partnerinstanzen zur Nachrichtenannahme und -weitergabe aus. Die mindestens eine den mindestens einen weiteren Prozeß enthaltende weitere Instanz leitet diese Nachricht zu wenigstens einer von ihr adressierten Serverinstanz weiter und erhält gegebenfalls von der wenigstens einen Serverinstanz eine Nachricht zur Weiterleitung über die den ersten Prozeß enthaltende erste Instanz an die Clientinstanz.

Figur 1


Prozess des ObjectHandler1
Prozess des ObjectHandler2
Client
Confirm Action
ObjectHandler1
ObjectHandler2
Action
Server
invoke_action
(server_inst, action, information)
get handle
number
start timer,
find target
ObjectHandler
action_request
(server_inst,
action,
information)
execute
(server_inst,
information)
action
(information)
store
communication
partner
invoke_action (client_inst,
confirm_action, reply)
action_request
(client_inst,
confirm_action,
reply)
execute
(client_inst,
reply)
confirm_action
(reply)
release communication partner
release handle
number
stop timer

Figur 1


Caller
AMOI
SIBS
ABM
set_alarm
check_params
create contained
Alarm
Confirm
Balance SIBS
accumulate
alarm status of
all associated AFS
Balance ABM
accumulate
alarm status of
all associated SIBS

Figur 2